REMARKS

The specification has been reviewed and amendments have been made to correct minor grammatical and typographical errors. No new matter has been added.

Claims 1 to 15 remain in the application. By this amendment, claims 1, 4–7 and 9–15 have been amended.

The withdrawal of the previous Final rejection mailed May 25, 2005, is noted with appreciation.

Claims 1 to 15 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,301,709 to Warmink in view of U.S. Patent No. 4,823,256 to Bishop et al., newly cited by the Examiner. This rejection is respectfully traversed for the reason that the combination of Warmink and Bishop et al. does not suggest or otherwise teach the claimed invention.

The program updating system having a communication function according to the claimed invention is comprises a first processor 1 (Figure 2) which operates by referring to a program stored therein in flash ROM 1a and a second processor 2 which executes update of the program via communications buffer 5 by using a communication function with an external unit, and executes an update control of the program when a fault of the first processor 1 is detected. In this program updating system, the second processor 2 transmits periodic reset signals to the first processor 1 and then monitors a response pulse Pa which is transmitted from the first processor 1 in response to the reset signal. The second processor 2 transmits a compulsory reset signal (La : Low) to the first processor 1 when the response pulse can not be detected within a predetermined period. The program updating system may further include an activation monitoring circuit or watch dog timer 13 (Figures 8 and 9) which generates an activation pulse to activate the second processor 2, and then monitors transmission of an activation response pulse which is outputted from the second processor 2 in response to the activation pulse. In this configuration, the activation monitoring circuit 13 transmits a compulsory reset signal to the second processor 2 when the activation response pulse can not be detected within the predetermined period. Further, the second processor 2 may

further include a buffer 14a (Figure 9) which transiently stores the program for executing the update control. In this configuration, the second processor 2 transfers the program stored in the buffer 14a to the first processor 1 after an operation of storing the program in the buffer 14a is completed.

Warmink discloses in Figure 1 a system 10 which includes a first circuit pack 20 a second circuit pack 30 and a means for communication between the circuit packs 50. The system also includes a hardware switch 60, a hardware switch signal line 70, a first reset 40, and a second reset 45. First circuit pack 20 contains first memory 90, and second circuit pack 30 contains second memory, 100. When either of the first circuit pack 20 or the second circuit pack 30 receives a reset signal from its respective reset line (first reset 40 for the first circuit pack 20 and second reset 45 for the second circuit pack 30), that circuit pack becomes the slave circuit pack, and acts in a specific way. If, for example, the first circuit pack 20 has received the signal and is the slave circuit pack, it sends a request over the means for communication 50 to second circuit pack 30 to send information from the second memory 100 about the version of data stored there. The second circuit pack 30 then becomes the master circuit pack. In response to the request, the second circuit pack 30 then sends to the first circuit pack 20 over means for communication 50 the version information requested. First circuit pack 20 compares the version information received from the second circuit pack 30 to version information it has internally in the first memory 90. It also checks the state of the hardware switch 60 by checking the signal on the hardware switch signal line 70. If the version information received by the first circuit pack 20 from second circuit pack 30 is more current than the version information stored in first memory 90 and if the hardware switch signal is on, then the first circuit pack 20 initiates an update of first memory 90. If either the version information received is older (or the same as) the version information stored, or the hardware switch signal is off, then no update will occur, and nothing further occurs until the next reset signal on one of the reset lines 40 and 45.

The Examiner states that "Although, Warmink doesn't disclose the second processor executing the update control of said program when a fault of said first processor is detected, he does mention any circuit pack in memory may need to be

upgraded at some point to fix bugs (errors in code)." Warmink does state at column 1, lines 15 and 16, that "software may need to be upgraded to fix bugs or to introduce new features"; however, there is nothing in the Warmink disclosure concerning the detection of "a fault of said first processor", as the Examiner alleges in his rejection. In fact, there is no monitoring of a first processor by a second processor to detect a fault in the first processor as specifically claimed.

Bishop et al. disclose a duel processor system 100 with duplicated memory 114, 124 which has two modes 10, 11 of operation: a converged mode 10 in which one of the two processors 101 or 102 is active and executing all system tasks while the other processor is inactive; and a diverged mode 11 in which both processors are active and independently executing different tasks. The system automatically changes modes in response to requests such as manual and program control and certain system fault conditions. In diverged mode, the system may be in either of two states of operation 1 and 2. In state 1, one processor 101 is designated a primary processor, and in the other state 2 the other processor 102 is designated the primary processor. In the converged mode the system may be in either of four states of operation, designated states 3–6. In states 3 and 4, one processor is active while the other processor is standing by ready to take up execution of tasks from the point where the one processor stopped execution. In states 5 and 6, one processor is active while the other processor is out of service and cannot take up task execution without being initialized. The system 100 makes transitions between the various states in response to requests. Except for transitions of an active processor to an out-of-service condition, the state transitions are transparent to tasks other than fault recovery programs and, upon a fault condition, the faulted program.

The Examiner states that "Bishop in an analogous art and similar configuration does disclose having and [sic] a first and second memory for a first and second processor wich is automatically updated during fault recovery (16:45 – 17:10)." The passages of Bishop et al. cited by the Examiner are as follows:

> "If the attempted soft change fails, the fault recovery software of the affected processor0 101 attempts a hard change to the other processor1 102. The type of hard change undertaken is a function of the current configuration of the system 100. If the

> system is in an active-active configuration, the attempted hard change is a hard converge to the other processor1 102. The process of hard convergence is discussed in conjunction with FIG. 13. If the system 100 is in an active-standby configuration, the hard change is a hard switch to the other processor1 102. The process of hard switching is discussed in conjunction with FIG. 14.
>
> "If the soft change is successful, the fault recovery program, which is now executing on the remaining active processor1 102, informs the operating system kernel of the soft change.
>
> "After informing the kernel of the soft change, or if updating of a counter did not result in its exceeding a specified threshold count, the fault recovery program examines the information that has been gathered about the fault to determine whether the software of the system 100 has been affected by the fault. If software is not affected, the processor on which the kernel is executing resumes execution of system tasks. If software is affected, the fault recovery program also makes available to the kernel the information that has been gathered about the fault.
>
> "As the above discussion indicates, the kernel that is provided information about the fault is executing either on the faulted processor0 101 or on the other processor1 102, depending on whether or not a change of processors was successfully made. The kernel examines the information received from the fault recovery program to determine whether the kernel itself is affected by the fault, or whether only other, non-kernel, programs are affected thereby."

What is described by Bishop et al. is a fault recovery system which relies on the redundancy of the two processors. While Warmink and Bishop et al. have in common certain redundancy schemes, this is not what is being claimed.

Claim 1 recites "A *computer firmware program updating system having a communication function* comprising: a first processor which operates by referring to a firmware program stored in a flash ROM; and a second processor *which monitors an operation of said first processor and executes update of said firmware program by using said communication function with an external unit,* and executes an update control of said program *when a fault of said first processor is detected*" (emphasis added). This is fundamentally different than either Warmink or Bishop et al. and the combination of the two would not produce the claimed invention. Neither of the references suggests the concept of monitoring a first processor by a second processor to detect a fault and updating, or replacing, the firmware of the first processor via a communication function with an external

unit. The claimed invention is not a redundant system in which one processor is substituted for the other; rather, the program updating system according to the claimed invention includes second processor for controlling the update of the program and the first processor for executing the program. If the first processor cannot carry out the normal response to the second processor within a predetermined period, the operation of the first processor is compulsorily stopped. This makes it possible to avoid the fault caused by the runaway operation of the first processor, and even in the case of the stop of the operation of the first processor, the process for updating the firmware can be executed under control of the second processor.

Claim 2 is dependent on claim 1 and adds that "said second processor transmits a reset signal to said first processor for every predetermined number of cycles, and monitors a response pulse which is transmitted from said first processor in response to said reset signal, and transmits a compulsory reset signal to said first processor when said response pulse can not be detected within a predetermined period". This is monitoring and fault detection function carried out by the second processor. No such operation is suggested by the combination of Warmink and Bishop et al.

Claim 3 is dependent on claim 2 and adds "an activation pulse generating circuit which generates an activation pulse to activate said second processor, wherein said second processor starts transmitting said reset signal in response to said activation pulse outputted from said activation pulse generating circuit." This is the power-on reset circuit 3 of Figure 2 or, alternatively, the watch dog timer 13 of Figures 8 and 9.

Claim 4 is dependent on claim 3 and adds "a buffer which transiently stores a program to be used for executing said firmware program update, wherein said second processor transfers said program stored in said buffer to said first processor for writing into said flash ROM, after an operation of storing said program in said buffer is completed." This buffer is shown, for example, as EEPROM buffer 14a in Figure 9.

Figure 5 is dependent on claim1 and adds the limitations of claim 3, while claim 6 is dependent on claim 5 and adds the limitations of claim 4. Claim 7 is

dependent on claim 1 and adds the limitations of claim 4. Claim 8 is dependent on claim 2 and adds limitations similar to claim 5, while claim 9 is dependent on claim 8 and adds the limitations of claim 4. Claim 10 is dependent on claim 1 and adds the limitations of claim 8, while claim 10 is dependent on claim 10 and adds the limitations of claim 4. These dependent claims are patentable over the combination of Warmink and Bishop et al. for the reasons advanced above.

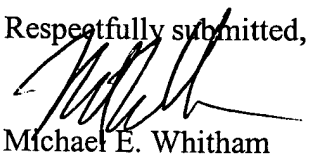
Claim 12 is directed to a firmware updating method and is similar in scope to claim 2. This claim, and dependent claims 13–15 are also patenatable over the combination of Warmink and Bishop et al. for the reasons advanced above.

In view of the foregoing, it is respectfully requested that the application be reconsidered, that claims 1–15 be allowed, and that the application be passed to issue.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

A provisional petition is hereby made for any extension of time necessary for the continued pendency during the life of this application. Please charge any fees for such provisional petition and any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 50-2041.

Respectfully submitted,



Michael E. Whitham
Reg. No. 32,635

Whitham, Curtis & Christofferson, P.C.
11491 Sunset Hills Road, Suite 340
Reston, VA 20190

Tel. (703) 787-9400
Fax. (703) 787-7557

Customer No.: 30743